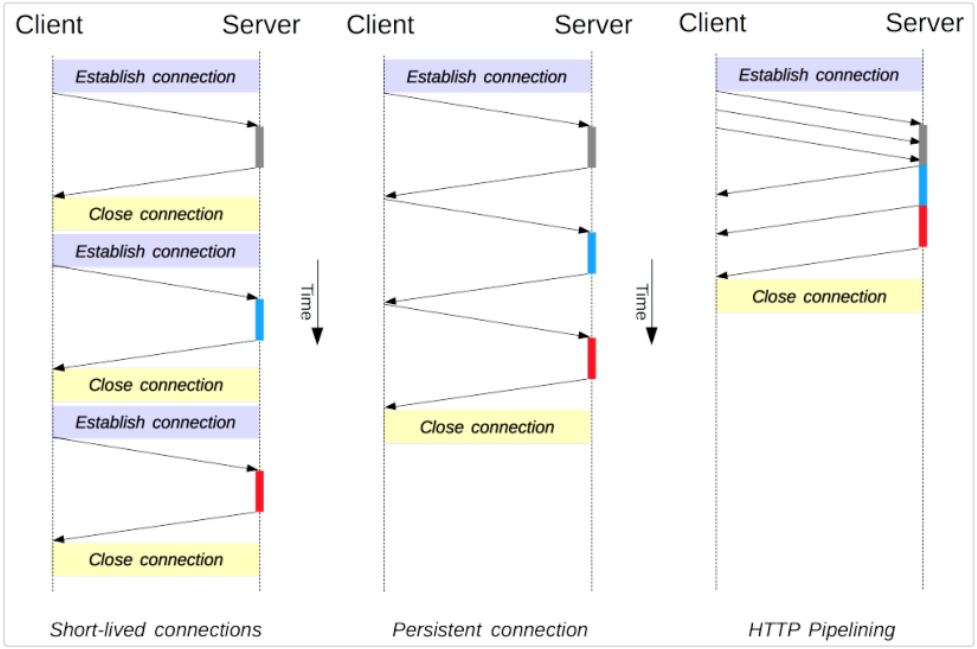
Gerenciamento de conexão no HTTP/1.x

O gerenciamento de conexão é um tópico-chave no HTTP: abrir e manter conexões impacta fortemente o desempenho de sites e aplicações web.
No HTTP/1.x, existem vários modelos: conexões de curta duração, conexões persistentes e *HTTP pipelining*.

O HTTP baseia-se principalmente no TCP como seu protocolo de transporte, fornecendo uma conexão entre cliente e servidor.
Em seus primórdios, o HTTP usava um único modelo para lidar com essas conexões.
Essas conexões eram de **curta duração**: uma nova era criada toda vez que uma requisição precisava ser enviada, e era encerrada assim que a resposta era recebida.

Esse modelo apresentava uma limitação inata de desempenho: abrir cada conexão TCP é uma operação que consome recursos.
Várias mensagens precisam ser trocadas entre cliente e servidor. A latência da rede e a largura de banda afetam o desempenho quando uma requisição precisa ser enviada.
Páginas web modernas exigem muitas requisições (uma dúzia ou mais) para entregar a quantidade de informações necessária, o que prova que esse modelo inicial é ineficiente.

Dois modelos mais recentes foram criados no HTTP/1.1.
O modelo de **conexão persistente** mantém as conexões abertas entre requisições sucessivas, reduzindo o tempo necessário para abrir novas conexões.
O modelo de **HTTP pipelining** vai um passo além, enviando várias requisições sucessivas **sem sequer esperar por uma resposta**, reduzindo muito da latência da rede.



É importante observar que o gerenciamento de conexão no HTTP se aplica à conexão entre **dois nós consecutivos**, o que é **hop-by-hop** (salto a salto) e não **fim-a-fim**.
O modelo usado entre um cliente e seu primeiro proxy pode diferir do modelo entre esse proxy e o servidor de destino (ou quaisquer proxies intermediários).
Os cabeçalhos HTTP envolvidos na definição do modelo de conexão, como Connection e Keep-Alive, são **cabeçalhos hop-by-hop**, com valores que podem ser modificados por nós intermediários.

Nota: o HTTP/2 adiciona modelos adicionais para gerenciamento de conexão.

Conexões de curta duração

O modelo original do HTTP, e o padrão no HTTP/1.0, é o de conexões de curta duração.
Cada requisição HTTP é concluída em sua própria conexão; isso significa que ocorre um *handshake* TCP antes de cada requisição HTTP, e essas conexões são serializadas.

O *handshake* TCP em si consome tempo, mas uma conexão TCP se adapta à sua carga, tornando-se mais eficiente com conexões mais sustentadas (ou "aquecidas").
Conexões de curta duração **não aproveitam** esse recurso de eficiência do TCP, e o desempenho se degrada por persistirem em transmitir por uma nova conexão fria.

Esse modelo é o padrão usado no HTTP/1.0 (**se não houver cabeçalho Connection ou se seu valor for close**).
No HTTP/1.1, esse modelo só é usado quando o cabeçalho Connection é enviado com o valor close.

Conexões persistentes

As conexões de curta duração têm duas grandes falhas:

- o tempo necessário para estabelecer uma nova conexão é significativo,
- o desempenho da conexão TCP só melhora após ela estar em uso por algum tempo (conexão "aquecida").

Para atenuar esses problemas, o conceito de conexão persistente foi criado, **mesmo antes do HTTP/1.1**.
Alternativamente, isso pode ser chamado de conexão *keep-alive*.

Uma conexão persistente é aquela que **permanece aberta por um período de tempo** e pode ser reutilizada para várias requisições, poupando a necessidade de um novo *handshake* TCP e utilizando os recursos de melhoria de desempenho do TCP.
Essa conexão **não permanecerá aberta indefinidamente**: conexões ociosas são fechadas após algum tempo (um servidor pode usar o cabeçalho Keep-Alive para especificar um tempo mínimo que a conexão deve permanecer aberta).

Conexões persistentes também têm desvantagens: mesmo quando ociosas, consomem recursos do servidor, e sob carga intensa, **ataques DoS** podem ser realizados.
Nesses casos, o uso de conexões **não persistentes**, que são encerradas assim que ficam ociosas, pode proporcionar melhor desempenho.

As conexões no HTTP/1.0 **não são persistentes por padrão**.
Definir o cabeçalho Connection com qualquer valor diferente de close (geralmente keep-alive) as torna persistentes.

No HTTP/1.1, **a persistência é o padrão**, e o cabeçalho nem sempre é necessário (embora muitas vezes seja adicionado como medida de segurança contra retrocompatibilidade com o HTTP/1.0).

HTTP pipelining

O *HTTP pipelining* leva a ideia de conexões persistentes um passo adiante: permite que **várias requisições** sejam feitas em uma única conexão TCP **sem esperar pela resposta da anterior**.

Em uma conexão HTTP/1.1 persistente sem pipelining, o cliente deve esperar que a resposta de cada requisição chegue **antes de enviar a próxima**.
O pipelining permite enviar várias requisições **de uma vez só**, de forma serial, e então aguardar todas as respostas.

Essa técnica **melhora muito o desempenho percebido**, especialmente em conexões de alta latência (como em redes móveis ou satélite).

No entanto, o pipelining possui desvantagens consideráveis:

- O servidor **deve enviar as respostas na mesma ordem** em que as requisições foram recebidas.
  Isso significa que se uma resposta demorar para ser processada, **as seguintes serão bloqueadas**, mesmo que estejam prontas.
- O pipelining **não é amplamente suportado**. Muitos servidores e proxies **não o implementam corretamente**.
- Ele **não lida bem com erros de conexão**. Se uma requisição intermediária falhar, as subsequentes podem ser comprometidas.

Devido a essas limitações, o pipelining **nunca foi ativado por padrão** na maioria dos navegadores modernos (com exceção do Opera).
Ele é considerado um mecanismo experimental e caiu em desuso com a chegada do **HTTP/2**, que lida com esse problema de forma mais robusta por meio de **multiplexação**.

Domain sharding

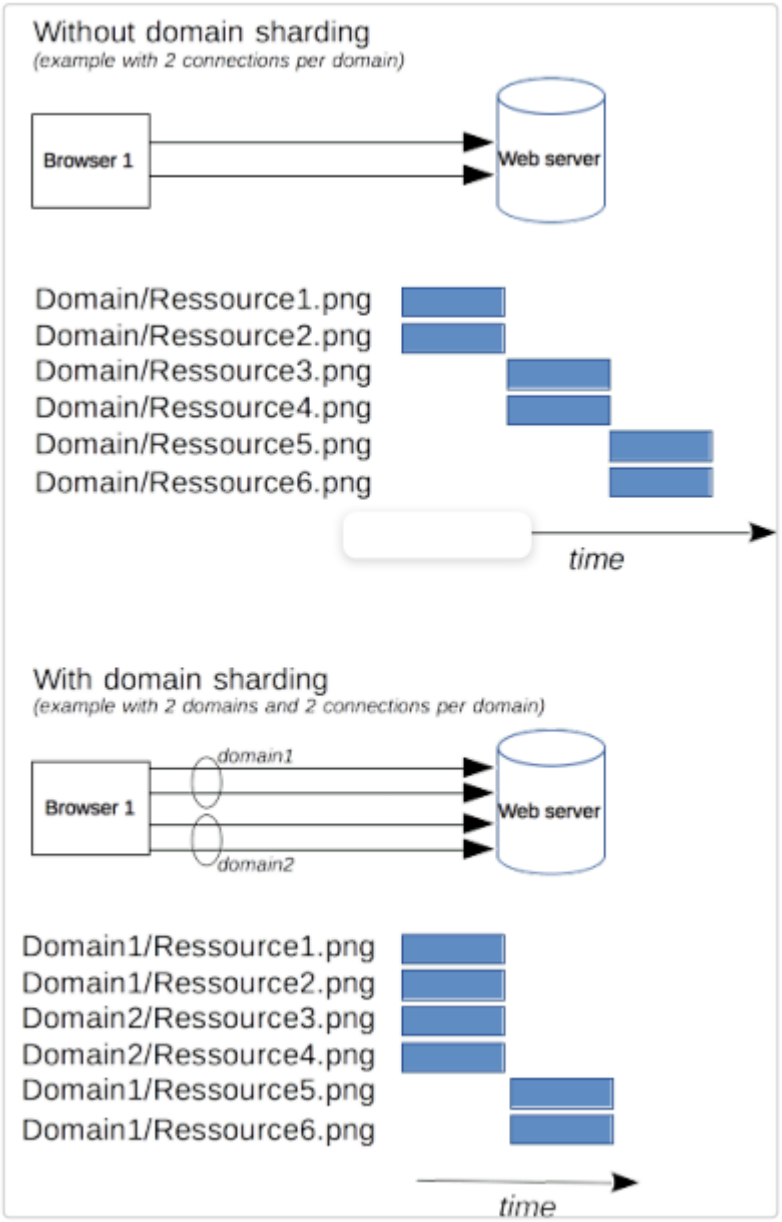
Fragmentação de domínios

Para **melhorar o desempenho percebido** com o HTTP/1.x, os navegadores **limitam o número de conexões simultâneas** que podem ser abertas com o mesmo domínio (geralmente 6 conexões por domínio).
Se uma página exigir o download simultâneo de muitos recursos, isso pode **criar um gargalo**.

Para contornar essa limitação, sites costumavam utilizar **múltiplos subdomínios** (como img1.exemplo.com, img2.exemplo.com) que apontavam para o **mesmo servidor**.
Isso permitia **contornar o limite de conexões por domínio**, já que os navegadores tratavam cada subdomínio como um destino separado, abrindo mais conexões em paralelo.

Esse truque é conhecido como **domain sharding** (fragmentação de domínios) e era uma técnica comum de otimização no HTTP/1.x.

Nota: no **HTTP/2**, esse truque **não é mais necessário**, pois o protocolo suporta **multiplexação** de múltiplas requisições/respostas simultâneas **em uma única conexão**.
Na verdade, o uso de domain sharding pode **prejudicar** o desempenho em HTTP/2, ao forçar múltiplas conexões que poderiam ser consolidadas.



Detecção de suporte a conexões persistentes

Os navegadores modernos geralmente mantêm conexões persistentes por padrão.
O suporte a conexões persistentes pode ser determinado pelo valor do cabeçalho Connection.

Protocolo	Cabeçalho de requisição	Cabeçalho de resposta	Persistência
HTTP/1.0	—	—	Não
HTTP/1.0	`Connection: keep-alive`	`Connection: keep-alive`	Sim
HTTP/1.1	—	—	Sim
HTTP/1.1	`Connection: close`	`Connection: close`	Não

Nota: O uso do cabeçalho Connection: keep-alive em HTTP/1.1 é tecnicamente desnecessário,
mas frequentemente é mantido por motivos de compatibilidade com HTTP/1.0.